Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vid Gabrijel

Razvoj modula za nadzor in analizo stroškov poslovanja

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Branko Šter

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Razvijte in implementirajte modul spletne aplikacije za nadzor in analizo stroškov poslovanja. Modul naj bo prilagojen potrebam manjših podjetij. Opišite uporabljene spletne tehnologije, strukturo modula, njegovo delovanje, uporabniški vmesnik in vrste analize stroškov, ki jih modul omogoča.

Rad bi se zahvalil mentorju prof. dr. Branku Šteru za vso strokovno pomoč ter nasvete pri načrtovanju in izvedbi diplomske naloge. Posebna zahvala pa gre bratu Elvinu Gabrijelu, sestri Jasmini Jevnikar, materi Nevenki Gabrijel ter prijatelju Mitji Stovanje za moralno podporo ob izdelavi diplomske naloge.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hyper Text Markup Language	Jezik za označevanje nadbesedila
PHP	Personal Home Page Tools	Orodja za osebno spletno stran
AJAX	Asynchronous JavaScript and XML	Asinhroni JavaScript in XML
XML	Extensible Markup Language	Razširljivi označevalni jezik
WWW	World Wide Web	Svetovni splet
CSS	Cascading Style Sheets	Kaskadne stilske predloge
API	Application programming interface	Vmesnik za dostop do sistemskih funkcij
DOM	Document Object Model	Objektni model dokumenta
SQL	Structured Query Language	Strukturirani povpraševalni jezik
ISAM	Indexed sequential access method	Metoda za indeksirane dostope
VSAM	Virtual Storage Access Method	Metoda za dostop do virtualnega shranjevanja
BSD	Berkeley Software Distribution	Distribucija programske opreme Berkeley
DSL	Domain-specific language	Domeni specifični jezik
RBAC	Role-based access control	Nadzor dostopa na podlagi vloge
LBAC	Lattice-based access control	Nadzor dostopa na osnovi rešetke

Povzetek

Naslov: Razvoj modula za nadzor in analizo stroškov poslovanja

Avtor: Vid Gabrijel

V diplomskem delu obravnavamo razvoj ter implementacijo modula za nadzor in analizo stroškov poslovanja. Danes se veliko manjših podjetij namreč ne odloči za implementacijo informacijskega sistema namenjenega nadzoru poslovanja, čeprav lahko ta močno pripomore k izboljšanju finančnega poslovanja. Razloga za strah pred vpeljavo novodobnih sistemov sta običajno cenovna zahtevnost implementacije in zahtevna uporaba programov, saj so ponudbe v osnovi namenjene uporabi večjih podjetij. Potrebno je razlikovati med potrebami velikih in manjših podjetij – manjša podjetja potrebujejo svojemu delovanju prilagojene programe. V ta namen bomo skozi diplomsko nalogo spoznali razvoj modula, ki je namenjen uporabi v manjših podjetjih, ter jim tako zagotoviti preprosto, vendar učinkovito izboljšanje poslovanja. Opisali bomo, kako na eleganten način omogočamo uporabniku dostop do aplikacije in do uporabe le tistih modulov, ki jih uporabnik potrebuje, ter ga z njemu nepotrebnimi elementi sistema ne obremenjujemo. Poudarili pa bomo seveda tudi osnove poslovnega finančnega poslovanja ter se sprehodili skozi delovanje modula in si pogledali, kako uporabnik skozi uporabniški vmesnik manipulira z opravljanjem stroškov.

Ključne besede: modul, razvoj, stroški, analiza, spletna aplikacija, Twig, JavaScript, PHP.

Abstract

Title: Module development for control and analysis of business costs

Author: Vid Gabrijel

The main purpose of the bachelor's thesis at hand is the presentation of the development and implementation of a module intended to control and analyse operating costs. Nowadays many small businesses remain reluctant to implementation of an information system for the control over the company's operations, notwithstanding the fact that the mentioned systems lead to improvement of the financial management. The main reasons for reluctance are usually a high implementation price and very demanding use of the programs, as the programs on the market suit the needs of larger businesses. It should be stress that there is a difference between large and small businesses – small businesses need programs suitable to their activities. Therefore, the thesis at hand presents the development of the module provided to suit the use of smaller businesses. The thesis describes how in an elegant way we allow the user to access the application and to use only those modules that the user needs and is not burdened with unnecessary elements of the system. In addition, the thesis at hand pays attention to the basis of the financial management, describes the detailed working of the module and demonstrates how the user can manipulate the costs via the user interface.

Keywords: module, development, costs, analysis, web application, Twig, JavaScript, PHP.

Poglavje 1

Uvod

Živimo v obdobju razvoja računalniških in informacijskih tehnologij, ki močno pripomorejo k razvoju na področju gospodarstva, zdravstva, gospodinjstva, šolstva ter mnogih drugih.

Pridobitev novih tehnologij in znanj na različnih področjih opazno spreminja pristop do dela ter tudi posledično na slog našega življenja. Na področju zdravstva lahko s pomočjo natančnih robotov opravimo operativne posege, ki so bili prej skoraj nemogoči, z računalniško analizo pa hitreje odkrivamo bolezni. V gospodarstvu stroji opravljajo natančna dela in pospešujejo proizvodnjo, s pomočjo informatike pa kontroliramo poslovanje ter finančno uspešnost podjetja.

Uporaba informacijskih tehnologij pri poslovanju je zelo pomembna, saj nam ponuja vpogled v poslovanje podjetja v realnem času, nadzor nad uspešnostjo zaposlenih ter preprosto izdelavo analize poslovanja, ki nam da pregled nad uspešnostjo podjetja. Z analizo poslovanja lahko vidimo, kateri stroški poslovanja so največje breme podjetju in kateri bodo v daljšem obdobju prinesli dobiček.

Kljub vsem tem prednostim pa veliko manjših podjetij še ne uporablja informacijskih tehnologij za nadzor nad poslovanjem podjetja, saj bi to pomenilo potrebo po učenju informacijskega sistema. Sistemi, ki jih uporabljajo večja podjetja, pa so zapleteni za integracijo in uporabo, saj ponujajo veliko

funkcij, ki jih manjša podjetja ne potrebujejo in so za uporabnika moteča. To pomeni, da v takih podjetjih veliko časa porabijo za pripravo predračunov, računov in drugih dokumentov, ki jih je treba nato še dostaviti računovodji.

V tem diplomskem delu bomo reševali omenjene težave z razvojem modula za nadzor in analizo stroškov poslovanja. Glavni cilj dela je izdelati spletno aplikacijo, dostopno uporabniku, vsakič ko ima omogočeno internetno povezavo. Služila bo kot informacijski sistem za podjetnike, ki za nadzor nad poslovanjem podjetja porabijo preveč časa, ki bi ga lahko investirali za delo, za katerega so specializirani. Pri izdelavi modula stremimo k prijazni uporabniški izkušnji, preprostemu vnosu stroškov podjetja in analizam, s katerimi omogočimo uporabniku prijazen pregled nad stroški ter statistiko poslovanja.

Diplomsko delo je razdeljeno v tri glavna poglavja – v 2. poglavju bodo opisane uporabljene tehnologije in orodja, ki smo jih uporabili pri izdelavi modula. V 4. poglavju bomo opisali, kako je modul zgrajen, prenos podatkov in sestavo baze, 5. – zadnje poglavje pa bo opisovalo analizo in prikaz podatkov, ki uporabniku na prijazen in preprost način prikažejo poslovanje podjetja v več želenih časovnih obdobjih.

Poglavje 2

Tehnologije in orodja

V tem poglavju bomo opisali tehnologije in orodja, ki so bila uporabljena pri razvoju modula.

2.1 Tehnologije

2.1.1 HTML

HTML (ang. *Hyper Text Markup Language*) je označevalni jezik za izdelavo spletnih strani. S CSS (ang. *Cascading Style Sheets*) in JavaScript-om oblikuje triado temeljnih tehnologij za WWW (ang. *World Wide Web*). Spletni brskalnik HTML dokument prejme s spletnega strežnika ali lokalnega repozitorija. HTML predstavlja osnovo spletnega dokumenta. Poleg prikaza dokumenta v spletnem brskalniku se z njim hkrati določi tudi semantični pomen delov dokumenta.

Izdela se ga lahko v vsakem urejevalniku besedil (v našem primeru PhpStorm), saj je zapisan v obliki elementov HTML, ki so sestavljeni iz značk, zapisanih v koničastih oklepajih. Značke HTML so običajno zapisane v parih (npr. $< b >, < /b >$) . Prva značka označuje začetek, druga pa konec veljave elementa. Značke lahko programer po lastni želji tudi gnezdi. Na Sliki 2.1 vidimo strukturo HTML dokumenta in uporabo značk.

example.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8
9  </body>
10 </html>
```

Slika 2.1: Osnovna struktura HTML dokumenta.

2.1.2 JavaScript

JavaScript je objektno skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani. Uporablja ga več spletnih mest, vsi moderni spletni brskalniki pa JavaScript podpirajo brez potrebe po vtičnikih, ki imajo vgrajen JavaScript interpreter (program oziroma interpreter, ki izvaja JavaScript kodo). Vsak med mnogimi JavaScript interpreterji predstavlja različno implementacijo JavaScripta, vse implementacije pa temeljijo na specifikacijah ECMAScript, s tem da nekatere specifikacij ne podpirajo popolnoma.

JavaScript podpira več slogov programiranja, kot sta na primer dogodkovni oziroma *event-driven* (delovanje programa je odvisno od dogodka, kot je na primer klik na gumb) in funkcijski oziroma »functional« (programiranje poteka z izrazi ali deklaracijami namesto izjavami). JavaScript ima tudi API za delo s besedilom, tabelami ter datumi.

JavaScript se veliko uporablja za ustvarjanje dinamičnih spletnih strani. Program se vgradi ali pa vključi v HTML (Slika 2.2), da opravlja naloge, ki niso mogoče samo s statično spletno stranjo (npr. odpiranje novih oken, izvajanje akcij s kliki na gumb, preprosti izračuni itd.).

example.html

```
<!DOCTYPE html>
<html>
<body>

<button type="button"
        onclick="document.getElementById('example').innerHTML = Date()">
    Prikaži čas</button>

<p id="example"></p>

</body>
</html>
```

Slika 2.2: Koda v JavaScript-u, vključena v HTML dokument, za prikaz trenutnega časa ob kliku na gumb.

2.1.3 jQuery

JQuery je odprtokodna JavaScript knjižnica, ki je namenjena posplošitvi komunikacije med HTML dokumentom oziroma DOM (ang. *Document Object Model* – API, ki elemente v dokumentih HTML (Slika 2.3), XHTML ter XML obravnava kot drevesno strukturo) in JavaScriptom. JQuery razvijalcem ponuja zmožnosti, da ustvarijo abstrakcije za nizkonivojske interakcije in animacije ter napredne učinke za visokonivojske tematske pripomočke. Modularni pristop h knjižnici jQuery omogoča razvoj močnih dinamičnih spletnih strani in aplikacij.

Filozofija jQuery-ja je *Napiši manj, naredi več* (ang. *Write less, do more*) [5]. Omenjeno filozofijo je mogoče razbiti na tri ideje:

- Poišči želeni element (npr. CSS element) in z njim upravljaj (npr. preko jQuery metode). Natančneje poišči zbirko elementov v DOM ter nato s to zbirko manipuliraj. Do tega pride, ko želimo prikaz nekega elementa z določeno vrednostjo uporabniku skriti, če nima dovoljenja za vpogled do tega določenega elementa (npr. `jQuery('div').hide()`, `jQuery('div').show()`).

```
17  <body>
18
19   <p>Example</p>
20
21  <script>
22      $( "p" ).wrap( "<div></div>" );
23  </script>
24
25  </body>
26  </html>
```

Slika 2.3: Uporaba jQuery knjižnice znotraj HTML.

- Veriženje več jQuery metod v niz elementov. jQuery je konstruiran tako, da dovoli veriženje jQuery metod. Veriženje nam prihrani čas pri pisanju kode ter ponuja lepšo preglednost nad njo (npr. `jQuery('div').hide()`, `jQuery('div').text('besedilo')`, `jQuery('div').show()` lahko lepše napišemo: `jQuery('div').hide().text('besedilo').show()`).

- Uporaba ovojnice jQuery in implicitne ponovitve. Z drugimi besedami, gre za izbiro DOM elementa s HTML strani, ki bo zavit (angl. wrapped) s funkcionalnostjo jQuery-a. V določenih primerih ovojnica vsebuje en DOM element, spet drugič jih vsebuje več.

2.1.4 AJAX

AJAX [1], kratica za asinhroni JavaScript in XML, je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij. Z AJAX-om si lahko spletne aplikacije izmenjujejo podatke s strežnikom asinhrono v ozadju (Slika 2.4), brez potrebe po ponovnem nalaganju strani. HTML in CSS se uporabljata za označevanje ter oblikovanje informacij, JavaScript pa uporabniku omogoča interakcijo s predstavljenimi informacijami. Podatki se prenašajo s pomočjo objektov XMLHttpRequest ali s pomočjo RemoteScriptinga v starejših brskalnikih.

Posledica AJAX-a je uporabniški vmesnik, ki se veliko hitreje odziva na

```
$.ajax({
        type: "POST",
        dataType: "json",
        url: "",
        data: {
                action : "deleteTable",
                id : _id,
        },
```

Slika 2.4: AJAX klic.

vnose uporabnika. Zmanjša se obremenitev spletnega strežnika, med odjemalcem in strežnikom pa se prenese veliko manj podatkov.

2.1.5 PHP

PHP [3] je razširjen, odprtokodni, splošno-namenski skriptni programski jezik, primeren za razvoj dinamičnih spletnih aplikacij, ki omogočajo interaktivnost. PHP-koda se lahko vključi v HTML-dokument ali pa se kliče v HTML-dokumentu iz posamezne datoteke (Slika 2.5). Temu dokumentu pravimo PHP-skripta in jo lahko izvajajo spletni strežniki s PHP-podporo, kot so Apache, IIS in drugi. Strežnik odpre skripto, s prevajalnikom izvede ukaze v njej, zapiše izpise v HTML, XML ali XHTML-dokument in ga pošlje odjemalcu. Tam ga obravnava spletni brskalnik in poskrbi, da se prikaže na zaslonu kot spletna stran.

Standardni PHP-interpreter, ki ga poganja Zed Engine (odprtokodni interpreter programskega jezika PHP), je brezplačna programska oprema, izdana v skladu z licenco PHP. PHP se lahko brezplačno uporablja na skoraj vseh spletnih strežnikih ter skoraj vsaki operacijski platformi.

V letih 2014 in 2015 je bila razvita nova različica PHP 7, zadnja izdana verzija pa je PHP 7.1.8. PHP 7 je uvedel nove jezikovne funkcije, vključno z deklaracijo *return type*, ki dopolnjujejo obstoječe deklaracije parametrov in podpirajo skalarne tipe (integer, float, string in boolean). PHP-interpreter

stroski.php

```php
<?php
include("");
include("");

$Site = new Site();
$payment_types = array();
$payment_types = $Site->get();


$assign=array(
    'ps'=>$payment_types,
    'ue'=>"0",
    'var'=>"Pregled načinov plačil"
);

$template= $twig->loadTemplate( name: '.html');
$template->display($assign);

?>
```

Slika 2.5: Uporaba PHP.

izvaja le kodo znotraj svojih oznak (`<?php koda?>`). Vse zunaj njegovih omejitev PHP ignorira.

Povezava na podatkovne baze jeziku PHP ne predstavlja težav, saj je povezljiv na več sistemov. V našem primeru se povezujemo na MySQL. Za povezavo na podatkovno bazo PHP ponudi funkcijo `mysql_connection()`. Funkcija potrebuje pet parametrov za vzpostavitev povezave ter vrne MySQL identifikacijsko povezavo o uspeli vzpostavitvi povezave. Povezavo z bazo prekinemo s PHP-funkcijo `mysql_close()`.

2.1.6 SQL

SQL ali strukturirani povpraševalni jezik za delo s podatkovnimi bazami. Je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkov-

nimi zbirkami uporablja programske stavke, ki posnemajo ukaze v naravnem jeziku. V primerjavi z starejšimi API-ji tipa »beri/piši«, kot sta ISAM ter VSAM, ima SQL dve glavni prednosti:

- Do več različnih elementov dostopa z enim samim ukazom
- Ne zahteva navedbe, kako dostopati do zapisa

.

```
    $sql="UPDATE '' SET '' ='".$fkid."', name='".$name."', amount='".$amount."',
    update_datetime=(CURRENT_TIMESTAMP) WHERE id = '".$id."';";


    $result=$db->query($sql);
}

function deleteCost(){
    global $db;

    $id = intval($_POST['id']);
    $sql="DELETE FROM '' WHERE id = '".$id."';";

    $result=$db->query($sql);
```

Slika 2.6: Primer uporabe SQL operacij DELETE in UPDATE.

SQL vključuje vnos podatkov, poizvedbe, posodobitve in brisanje, kreiranje in spreminjanje shem ter nadzor dostopa do podatkov. Jezik SQL je razdeljen na več jezikovnih elementov, vključujoč:

- izjave, ki lahko trajno vplivajo na shemo in podatke ali nadzorujejo transakcije, tok programa, povezave seje ali diagnostiko;
- poizvedbe, ki pridobivajo podatke na podlagi določenih meril. Je najbolj pogosta operacija v SQL-u, vrši pa se s stavkom SELECT. Stavek SELECT nima vpliva na vsebino tabele, vendar le vrne vsebino ene ali več tabel;

- izraze za ustvarjanje skalarnih vrednosti ali tabel, ki so sestavljene iz stolpcev in vrstic

Slika 2.6 prikazuje uporabo stavkov DELETE in UPDATE.

2.1.7 Bootstrap

Bootstrap, prvič poimenovan Twiter Blueprint, je brezplačno ter odprtokodno front-end orodje za oblikovanje spletnih strani ter aplikacij. Vsebuje predloge za tipografijo, oblike, gumbe in druge komponente vmesnika, ki temeljijo na HTML in CSS.

Bootstrap je modularen in je sestavljen iz vrste slogovnih predlog LESS, ki implementirajo več različnih komponent orodja. Te slogovne tabele so v spletno stran vključene kot zbirka v svežnju, vendar se lahko posamezne komponente doda ali odstrani.

Bootstrap ponuja nabor slogovnih predlog, ki zagotavljajo osnovne definicije za slog vseh ključnih komponent HTML. Ti ponujajo enoten, sodoben videz za oblikovanje besedila in tabel. Poleg običajnih elementov HTML Bootstrap vsebuje tudi druge elemente vmesnika, ki so pogosto uporabljeni. Komponente se implementirajo kot CSS razredi.

2.1.8 Twig

Twig [8] je odprtokodno ogrodno okolje za programski jezik PHP, licencirano pod licenco BSD (ang. *Berkeley Software Distribution*). Sintaksa Twiga izvira iz predlog Jinja in Djanga. Prvotno različico je ustvaril Armin Ronacher. Okvir Symfony PHP ima vgrajeno podporo za Twig kot svoj privzet interpreter predlog. Twig ima način »sandbox«, s katerim oceni varnost kode. To omogoča, da se Twig uporablja kot predlogni jezik za aplikacije, pri katerih lahko uporabniki spreminjajo obliko predloge.

Twig poganja fleksibilen »lexer« in »parser«. To razvijalcu omogoča, da po meri definira svoje lastne oznake in filtre ter ustvari lasten DSL (*domain-specific language*). Je napreden sistem, ki nam omogoča nadzor nad upo-

```php
$assign=array(
    'payment_types'=>$payment_types,
    'urejanje'=>"0",
    'var'=>"Pregled načinov plačil"
);

$template= $twig->loadTemplate( name: 'stroski.html');
$template->display($assign);

?>
```

Slika 2.7: Pošiljanje podatkov za izpis z pomočjo Twig-a.

```twig
{% for payment in payment_types %}
<tr>
    <td>{{ payment['id'] }}</td>
    <td>{{ payment['naziv'] }}</td>
    <td>{{ payment['vrstni_red'] }}</td>
    <td><a href="'?id={{ payment['id'] }}" id="urediBtn" class="btn btn-info">Uredi</a>
        <a href="#" data-id="{{ payment['id'] }}" class="btn btn-danger zbrišiBtn">Zbriši</a>
    </td>
</tr>
{% endfor %}
```

Slika 2.8: Izpis z pomočjo Twig-a.

rabniškim vmesnikom. V kombinaciji z YAMl (ang. YAML - *Ain't Markup Language*) služi kot močan ter enostaven sistem, ki je na voljo široki množici uporabnikov.

Twig deluje tako, da vse, kar je nepotrebno, izloči iz predloge. Predloge so zgolj besedilne datoteke, ki vsebujejo spremenljivke ali izraze, ki so nadomeščeni z vrednostmi. Oznake so pomemben del dokumenta, saj nadzorujejo logiko predloge. Twig ima dve glavni oznaki:

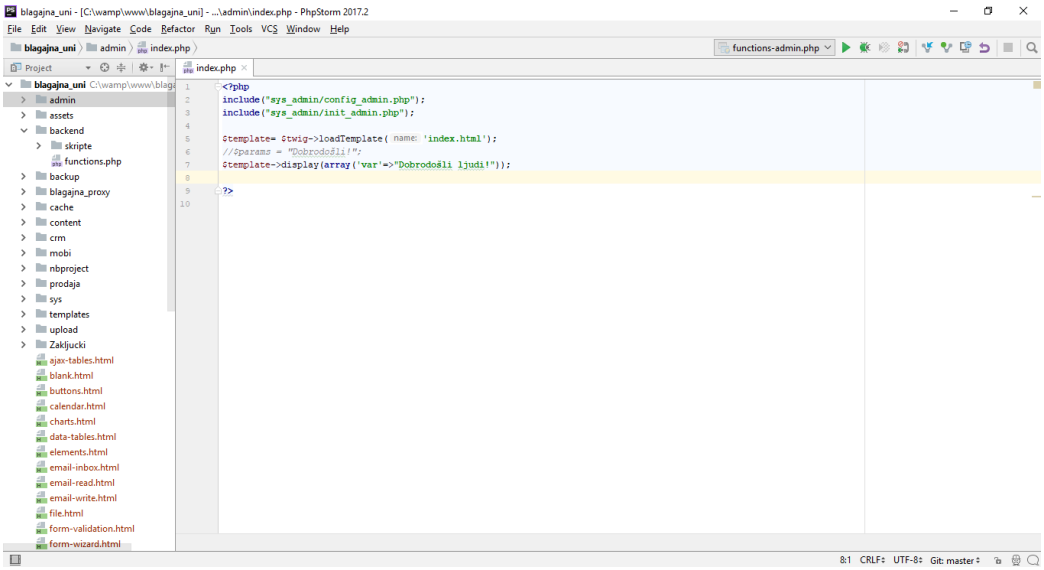
- {{}} izpiše rezultat ocenjevanja izrazov;
- {% %} izvršuje izjave.

Na Sliki 2.7 vidimo primer pošiljanja podatkov za izpis, na Sliki 2.8 pa vidimo še sam izpis za uporabo obeh glavnih oznak.

2.2 Orodja

2.2.1 JetBrains PhpStorm

JetBrains PhpStorm je komercialni, večplatformni IDE za PHP, ki temelji na platformi JetBrains' intelliJ IDEA (Slika 2.9). PhpStorm je urejevalnik za PHP, HTML ter JavaScript s sprotno analizo kode (ang. *on-the-fly code*; odkrivanje anomalij v kodi pred prevajanjem kode) in s samodejnim preoblikovanjem PHP ter JavaScript kode.



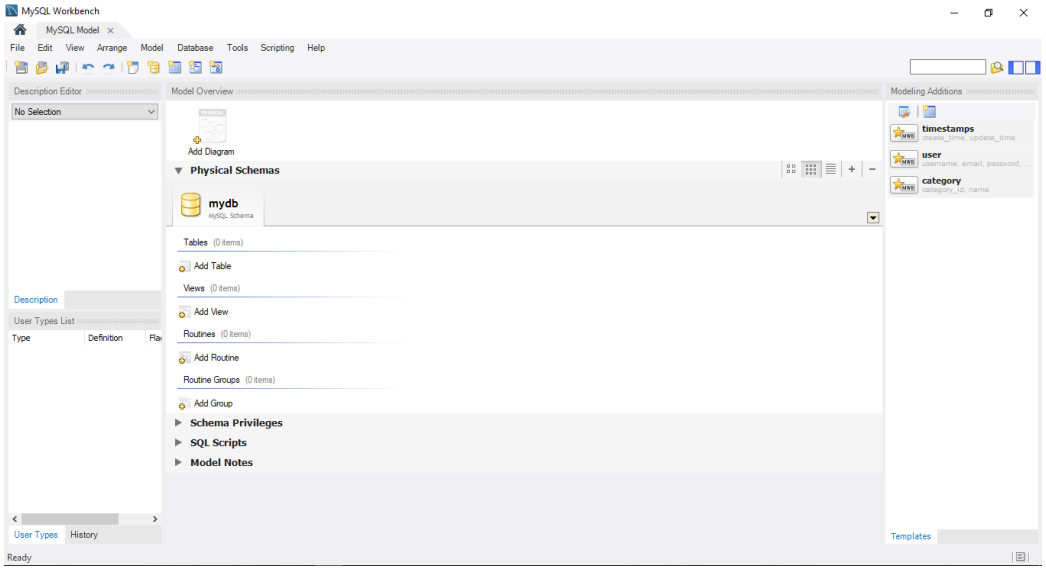
Slika 2.9: Urejevalnik PhpStorm.

2.2.2 WAMP

WampServer je spletno razvojno okolje, ki deluje na platformi Windows. Omogoča razvoj spletnih aplikacij z Apache2, PHP ter MySQL bazami. Za urejanje baze uporabljamo odprtokodno spletno aplikacijo phpMyAdmin.

2.2.3 MySQL Workbench

MySQL Workbench (Slika 2.10) je orodje za vizualno oblikovanje podatkovnih baz, razvito s strani Oracle Corporation, ki omogoča administracijo, oblikovanje podatkovnih baz, razvoj ter vzdrževanje podatkovnih baz v enotno integrirano razvojno okolje za podatkovni sistem MySQL. Orodje smo



Slika 2.10: Orodje MySql Workbench.

uporabili za razvoj ter načrtovanje podatkovne baze, ki smo jo nato izvozili v odprtokodno spletno aplikacijo phpMyAdmin.

Poglavje 3

Razvoj in uporaba modula

V tem diplomskem delu se bomo sprehodili skozi razvoj modula za vnos ter analizo stroškov poslovanja, ki bo namenjen manjšim podjetjem, ki nimajo informacijske podpore za vodenje poslovanja. Slednje dokazano zvišuje učinkovitost podjetij, saj lahko s pomočjo analiz načrtujejo svojo strategijo poslovanja ter vlaganja. V 3. poglavju Razvoj in uporabnost modula je predstavljeno, kako je potekal razvoj modula, ter dodatne vpeljave za efektivnost modula, ki ponuja poceni ter preprosto vpeljavo manjšim podjetjem, ki si prej takih sistemov niso mogla privoščiti. Na kratko pa bo modul predstavljen tudi z uporabniškega vidika.

3.1 Prijava v sistem

Ker se modul razvija za dodatek k že obstoječi aplikaciji, do katere ima dostop več uporabnikov, ki ne bodo imeli dostopa do modula za analizo stroškov, je bilo potrebno vpeljati sistem za prijavljanje uporabnika v program. Uporabniku bodo glede na odločitev »super Admina« (oseba, ki dodeljuje pravice in ima popoln pregled nad sistemom) dodeljene pravice, ki omejujejo uporabnika na uporabo aplikacije. Če je uporabniku dovoljen le vpogled v analizo poslovanja, po prijavi s svojim uporabniškim imenom ter geslom ne bo imel dostopa do ostalih elementov celotnega programa, ampak le do modula z ana-

lizami. V ta namen smo uporabili sistem »RBAC« (ang. *Role-based access control*)

3.1.1 *Role-based access control* (Rbac)

Role-based access control je metoda, ki regulira dostop do računalniških ali omrežnih virov na podlagi vlog posameznih uporabnikov v podjetju. To pomeni, da ima vsak posamezni uporabnik možnost, da izvede določeno nalogo, kot je ustvarjanje, ogled ali spreminjanje datoteke. Vloge so opredeljene glede na položaj uporabnika v podjetju. Na podlagi sistema RBAC se lahko spremlja vsakega uporabnika posamično ter je tako lažje spremljati učinkovitost zaposlenih ter nadzor napak v sistemu, saj je mogoče spremljati vse napake v poslovanju ter delovanju sistema.

Sistemu RBAC so dodeljena tri glavna pravila:

- Dodelitev vloge: Uporabnik lahko izvaja dovoljenje samo, če mu je bila dodeljena temu pravilu določena vloga;
- Dovoljenja vlog: Vloga uporabnika mora biti uporabniku dodeljena. Z zgornjim pravilom uporabniki lahko prevzamejo le vloge, za katere so pooblaščeni;
- Avtorizacija dovoljenj: Uporabnik lahko neko nalogo upravlja le v primeru, da je ta naloga dodeljena vlogi, ki je dodeljena uporabniku. Za to poskrbita 1. in 2. pravilo.

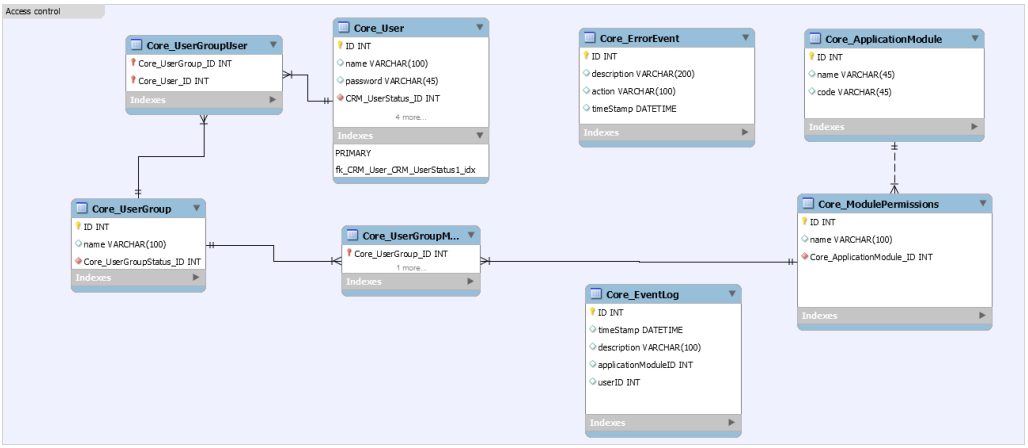
Dodatna pravila oziroma omejitve je možno v hierarhijo dodajati za bolj zanesljiv sistem.

3.1.2 Podatkovni model

V primeru vpeljave sistema RBAC v naš sistem smo vpeljali osnovno hierarhijo sistema RBAC. Za podporo pri nadaljnjem razvoju aplikacije smo dodali še opcijo za nadzor nad uporabniki, ki beleži izvedene akcije uporabnikov, ki so nato prikazane uporabniku, kateremu je dodeljeno dovoljenje za

vpogled v izvedene akcije (v večina primerih je to uporabnik, ki dodeljuje pravice »super Admin«). Omogočen je tudi nadzor nad izvajanjem aplikacije ter posameznimi izvedenimi akcijami za lažje odkrivanje napak v procesu delovanja aplikacije.

Naš podatkovni model (Slika 3.1) je tako sestavljen iz šestih glavnih tabel ter dveh dodatnih tabel za nadzor.



Slika 3.1: Podatkovni model

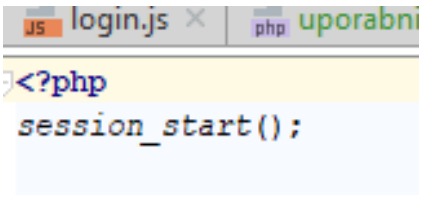
- Core_User: Vsebuje osebne podatke uporabnika, kontaktne podatke ter podatke za prijavo uporabnika v sistemu.
- Core_UserGroup: Vsebuje ime skupine, v katero je lahko dodeljen uporabnik.
- Core_UserGroupUser: Povezuje tabeli Core_User ter Core_UserGroup v strukturo tipa »mnogo-proti-mnogo«.
- Core_ApplicationModule: Tabela vsebuje elemente, ki bodo prikazani uporabniku le v primeru, da ima za to dodeljeno dovoljenje
- Core_ModulePermissions: Tu so zapisana vsa dovoljenja, ki bodo dodeljena uporabnikom (beri, urejaj, briši itd.).

- Core_UserGroupModulePermissions: Povezuje tabeli Core_UserGroup ter Core_ModulePermissions v strukturo tipa »mnogo-proti-mnogo«.
- Core_EventLog: Tabela za nadzor, v katero se shrani ime uporabnika, čas, opomba akcije ter modul v aplikaciji, nad katero je bila akcija izvedena.
- Core_ErrorEvent: Tabela za nadzor napak. V tabelo se zapiše akcija, opis napake ter čas, kdaj je do določene napake prišlo.

Načeloma ni omejitev pri dodajanju vlog ter dovoljenj. Razvijalec jih lahko ustvari, kolikor želi, da določi vloge in dovoljenje v zgornji organizaciji. Sami uporabniki možnosti registracije nimajo, saj le-te določi razvijalec. Uporabniki imajo le možnost manipulacije dostopnega gesla, kar pa se beleži ter je ob vsaki spremembi o tem obveščen tudi razvijalec. S koncepti hierarhije in omejitve vlog lahko uporabimo RBAC [6], da ustvarimo ali simuliramo *lattice-based access control* (LBAC). Tako je RBAC možno upoštevati kot nadgradnjo LBAC.

Zgornji sistem je v aplikacijo pripeljal veliko fleksibilnosti pri nadzoru uporabnikov ter delovanjem same aplikacije. Izboljšala se je tudi sama produktivnost zaposlenih, saj se je samo z nadzorom dostopov ter opravljenih akcij lažje vodilo zaposlene. Prav tako je z nadzorom napak postalo iskanje le-teh hitrejše ter učinkovitejše.

Za nadzor uporabnika mora biti le-ta prijavljen v aplikaciji, da imamo skozi njegov uporabni čas možnost pridobivanja podatkov o opravljenih akcijah. V našem primeru gre za spletno aplikacijo, kjer pa se pojavi težava, ker spletni strežnik ne ve, kdo je uporabnik, saj http ne shranjuje stanja. To težavo rešimo s spremenljivkami seje, v katere shranimo podatke o uporabniku (Slika 3.3), ki se v sistem prijavi, ter te podatke uporabljamo tudi na ostalih straneh. Vrednosti v spremenljivkah ostanejo, dokler se bodisi ne zapre brskalnik, se seja zaključi (`session_destroy()`) ali pa spremenljivke ročno poenostavimo (`session_unset()`). Za začetek seje uporabimo ukaz `session_start()` (Slika 3.2).



Slika 3.2: Začetek seje

```php
$email = $_POST['email'];
$pass = $_POST['pass'];
$sql="SELECT id FROM table_name WHERE user_pas = '".$pass."' AND email = '".$email."';";

$usrID=$db->query($sql);
if(is_null($usrID)){
    $usrID='';
}else{
    $_SESSION["userID"] = $usrID;
}
```

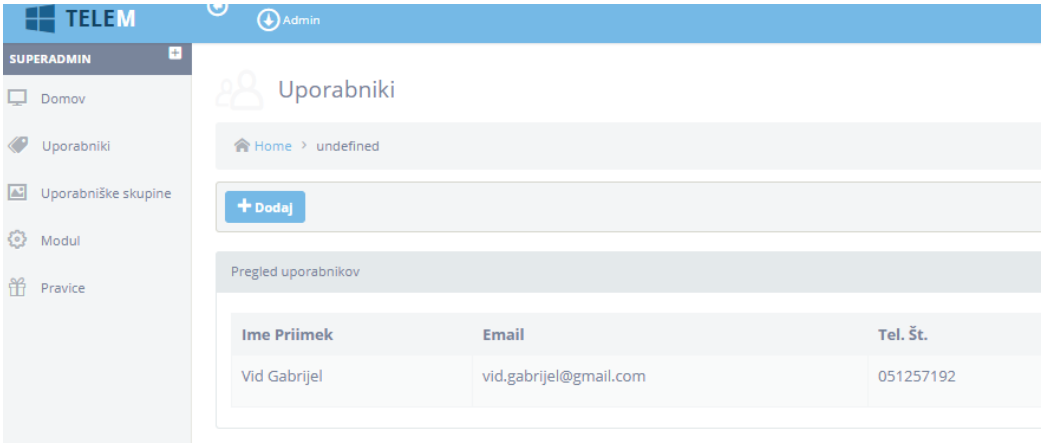
Slika 3.3: Shranjevanje uporabnikovega ID za nadalnjo uporabo

3.2 Vnos uporabnika in dodeljevanje pravic

Ker do same aplikacije nimajo dostopa vsi, ampak le zaposleni v podjetju, registracija v sistem ni omogočena. Uporabnika prijavi razvijalec, ki tudi določi, v katero uporabniško skupino spada uporabnik, ter mu tako dodeli pravice oziroma omeji dostop v aplikaciji.

Uporabnik lahko manipulira z uporabniškim računom tako, da ima možnost spremembe gesla, o tem pa je seveda obveščen tudi razvijalec oziroma »SuperAdmin«. Razvijalcu se v vmesniku (Slika 3.4) za dodajanje uporabnikov prikažejo že vsi vneseni uporabniki, katerim lahko razvijalec ureja osebne podatke ali pa jih izbriše iz sistema. Na samem vrhu vmesnika pa ima razvijalec možnost dodajanja novega uporabnika (npr. novi zaposleni), za kar se obrazec odpre na novi strani (Slika 3.5).

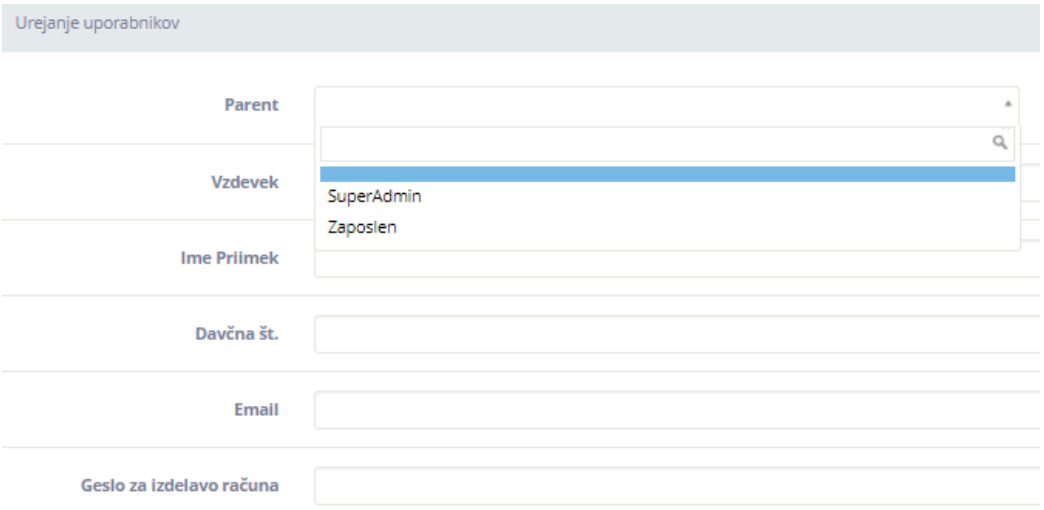
Pri vnosu uporabnika je potrebno poleg osebnih podatkov (ime in priimek, davčna številka itd.) izbrati tudi uporabniško skupino, v katero spada



Slika 3.4: SuperAdmin vmesnik

uporabnik (Slika 3.6). Uporabniško skupino je potrebno vnesti pred vnosom uporabnika. Prav tako je že pred tem potrebno določiti, kakšne pravice ta uporabniška skupina ima.

V postopku dodeljevanja pravic je potrebno pred tem določiti dele sistema, na katere se bodo pravice nanašale. Na primer aktualni modul, ki ga razvijamo skozi izdelavo diplomske naloge, za vnos ter analizo stroškov. V primeru, da se lastnik podjetja odloči, da tega modula ne vidi noben drug uporabnik razen njega, se mu dodeli edinstvena uporabniška skupina »SuperAdmin« (poimenovanje uporabniških skupin je prepuščeno lastni izbiri in ni vezano na delovanje sistema). Razvijalec vnese modul s stroški z identifikacijsko kodo »1« ter temu modulu dodeli vsa dovoljenja (Slika 3.6), ki jih želi omogočiti (tri osnovna so branje, pisanje ter brisanje). Nato razvijalec uporabniški skupini pripiše dovoljenja za uporabo modula s stroški. Tako naredi za vse module ter elemente sistema, pri katerih želi omejevati posamezne uporabniške skupine, ki uporabljajo sistem. Ko se bo lastnik vpisal s svojim uporabniškim imenom in geslom, se mu bodo prikazali vsi moduli, ki so mu bili dodeljeni. Po navadi se prikaže celotna struktura sistema. To se ne zgodi le v primeru, ko na primer lastnik ne bi želel imeti opravka z vnašanjem uporabnikov. V tem primeru se mu ne dodeli pravica za vpogled



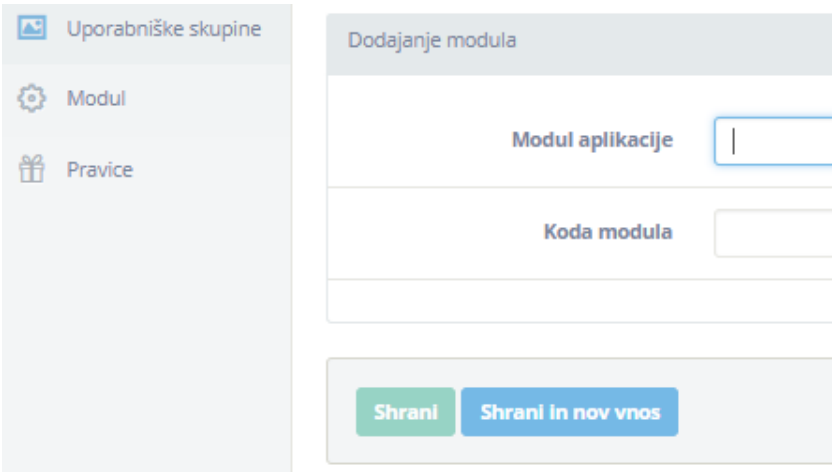
Slika 3.5: Izbira uporabniške skupine

modula za vnos uporabnikov ter ga tako ta del ne obremenjuje pri uporabi sistema za svoje delo. Tako bi pravice za urejanje uporabnikov dodelili drugi uporabniški skupini, ki bi bila za to zadolžena. Seveda pa lahko pride do primera v podjetju, ko dostopa do nekega določenega modula ne bo imela nobena uporabniška skupina oziroma dostopa ne bo potreboval. Tako uporabnikom omogočimo uporabo sistema na njemu prijazen način brez motečih elementov, ki jih ne potrebujejo za opravljanje svojega dela. Do sistema uporabnik dostopa preko obrazca, ki je prikazan na Sliki 3.7.

3.3 Stroški

V prvem delu drugega poglavja smo opisali, kako smo pri implementaciji poskrbeli za varnost z omejevanjem dostopov uporabnikom, ki jim določimo, do katerih modulov sistema smejo dostopati ter do kakšne mere lahko manipulirajo z vsebino sistema. Opisali smo, kako se te uporabnike vnese v sistem ter kako se jim dodeli pravice.

V drugem delu pa bomo opisali bistvo modula. Najprej bomo opisali teorijo stroškov, s katerimi upravljajo uporabniki modula, saj je to pri načrtovanju



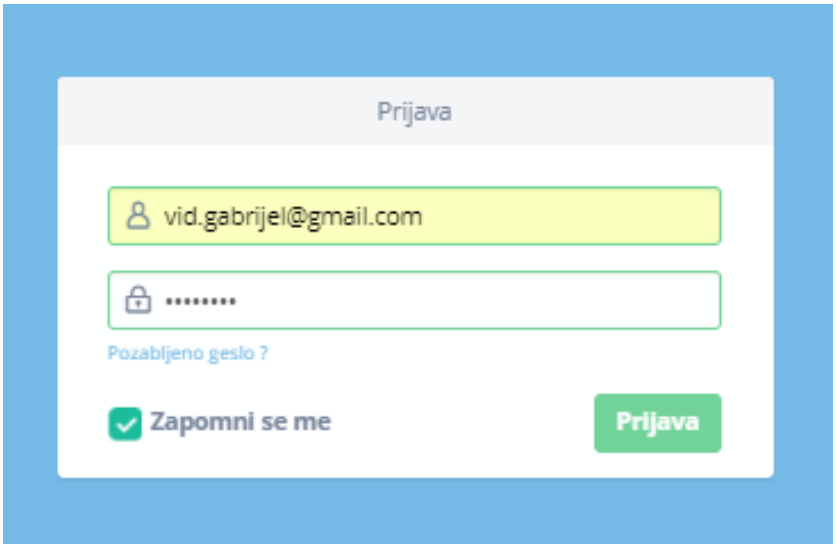
Slika 3.6: Obrazec za vnos modula kateremu se nato dodelijo pravice

modula zelo pomembno, ter nato še sam izgled vmesnika, ki uporabniku omogoča vnos teh stroškov.

3.3.1 Teorija stroškov

V poslovnih procesih uporabljamo osnovna sredstva, storitve, zaposlene ter material [2]. Stroški nam kažejo cenovno porabo sredstev poslovnega procesa. Stroške se razvršča v različne kriterije, ki pomagajo pri razumevanju metod vrednotenja proizvodov.

- Glede na kraj nastajanja: proizvajalni in splošni stroški (prodaja, nabava, uprava).
- Odvisnost od obsega poslovanja: stalni stroški (so stroški neodvisni od obsega proizvodnje. Sem spadajo stroški zavarovanja, obresti kreditov, najemnine, plače, ...) in variabilni stroški (stroški, ki so odvisni od obsega proizvodnje). Variabilni stroški se delijo v sorazmerno (stroški, ki kot celota naraščajo premo sorazmerno z obsegom poslovanja, kot povprečni pa ostajajo enaki), padajoče (stroški, ki kot povprečni in kot celotni stroški naraščajo hitreje kot obseg poslovanja) ter naraščajoče



Slika 3.7: Obrazec za prijavo uporabnika v aplikacijo

(stroški, ki kot celota in kot povprečje naraščajo počasneje kot obseg poslovanja) spremenljive stroške.

- Glede na povezanost z nastankom izdelka: neposredni in posredni stroški.

Ker bomo v modul razvijali predvsem za izvedo analiz poslovanja, je pomembno, da znamo podjetniku pomagati optimizirati davčno osnovo za plačilo davkov konec leta. Spodaj je predstavljenih osem vrst stroškov, na katere po navadi manjša podjetja pozabljajo, kar pa je podjetju lahko v veliko škodo.

- Stroški materiala: Stroški materiala so stroški osnovnega in pomožnega materiala ter kupljenih polproizvodov, delov, goriva, elektrike, pisarniški material ter surovine (stroški materiala, ki izhajajo iz kmetijstva, gozdarstva in rudarstva). Stroški materiala nastanejo šele po tem, ko iz polizdelka naredimo izdelek in po tem izdelek prodamo. Po tem procesu vidimo razliko v nabavi materiala za izdelavo izdelka ter vrednostjo, ki smo jo dobili za izdelek (kljub temu, da smo na začetku leta veliko investirali v nabavo materiala, se nam lahko do konca obračunskega leta

povrne). Izračun stroškov materiala v praksi ni preprost, saj se cene skozi čas spreminjajo.

- Stroški storitev: Poslovanja podjetja si ni mogoče zamisliti brez zunanjih storitev. Tukaj gre za stroške storitev najema poslovnih prostorov, stroške elektrike, službene poti, vode, ogrevanja, hlajenja ter interneta. Sem štejemo tudi stroške dajatev, ki niso odvisne od poslovnega izida in niso povezani s plačami ter s stroški obresti. Zunanjih storitev se v podjetju poslužujejo v primeru, da za opravilo določene storitve nimajo usposobljenega kadra ali pa opravljanje določenih nalog znotraj podjetja ne sovpada s poslovnim načrtom podjetja ter je tako podjetju boljša izbira zunanji ponudnik. Izračun stroškov storitev je preprost (količino storitev množimo z njihovo ceno), saj pri storitvah ni zalog.

- Strošek reprezentance: Za strošek reprezentance gre pri stroških za poslovne večerje, zabave ter darila, ki jih podarimo poslovnim partnerjem oziroma z nekom, za katerega domnevamo, da bo poslovni partner postal.

- Amortizacija: Amortizacija je opredeljena kot strošek, ki nastane zaradi prenašanja nabavne vrednosti osnovnega sredstva na poslovne učinke. Velikokrat pride do situacije, ko se ti stroški prenesejo v naslednje leto ter se za tekoče leto ne upoštevajo (nakup pisarniške opreme, prevoznih sredstev itd.).

- Rezervacije: Oblikovanje dolgoročnih rezervacij zmanjšuje dobiček obračunskega obdobja in posledično bilančni dobiček, o katerem odločajo lastniki [9]. S tem pa podjetje tudi zmanjšuje davčno osnovo in s tem vpliva na odmerjen davek, obveznost plačila davka pa prelaga na kasnejša obdobja. Rezervacije so potrebne za reorganizacijo, pričakovane izgube, jubilejne nagrade.

- Stroški obresti: Stroški obresti so stroški, ki se nanašajo na obračunane obresti od dobljenih posojil in drugih dolgov. So vse obresti, ki se

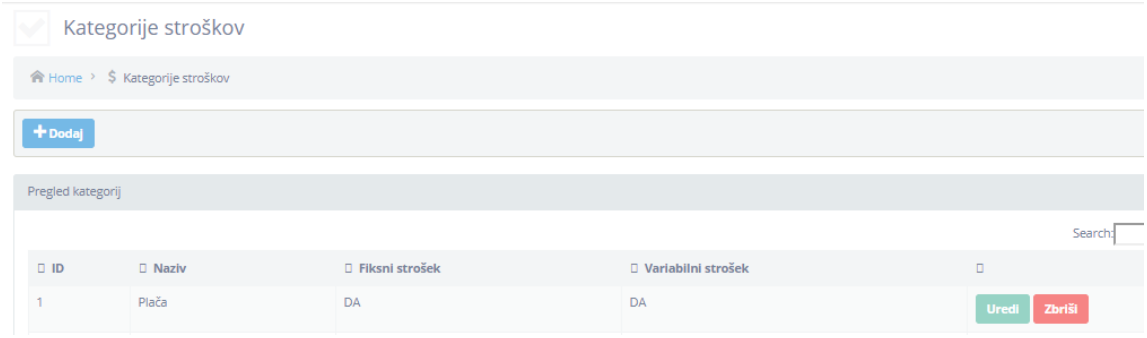
nanašajo na poslovanje. Če se nanašajo na nakupno, proizvajalno ali prodajno delovanje podjetja, se lahko kalkulantovo vštevajo tudi v lastno ceno poslovnih učinkov.

- Stroški dela: Stroški dela so vse oblike zaslužkov, ki jih daje organizacija zaposlenim (bruto plače, prevoz na delo, prehrana, regres ter vsi ostali stroški povezani z zaposlenimi). Tu se ne upoštevajo stroški potovanj in štipendij, saj se ti upoštevajo pri stroških storitev. Na podlagi stroškov dela se tudi regulira število zaposlenih v podjetju, saj moramo paziti, da te stroške pravilno razporejamo, da ne pride do izgube. Stroški podjetja morajo biti vedno usklajeni s prihodki ter obratno.

- Drugi stroški: To so stroški, ki jih ni mogoč opredeliti v katero iz med zgornjih kategorij (npr.: odpravnine, dokup delovne dobe, nagrade).

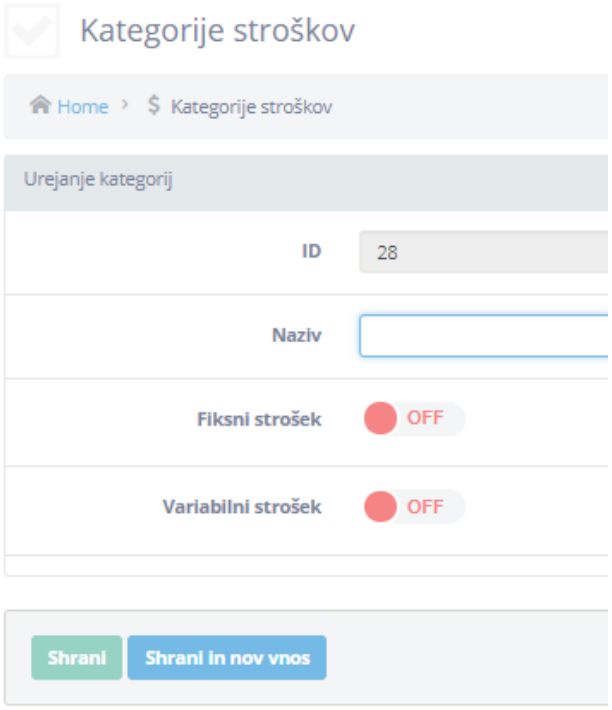
S poznavanjem stroškov, ki so davčno priznani, lahko podjetjem omogočimo, da jih na prijazen ter preprost način začnejo uveljavljati skozi uporabo naše spletne aplikacije ter tako optimizirajo stroške poslovanja.

3.3.2 Vmesnik za vnašanje stroškov



Slika 3.8: Vmesnik za prikaz kategorij stroškov

Uporabniku aplikacije moramo skozi uporabniški vmesnik omogočiti vnos stroškov, ki predstavljajo poslovanje uporabnikovega podjetja. Uporabnik



Slika 3.9: Obrazec za vnos kategorije stroška

stroške vnese v dveh korakih. Naprej vnese kategorijo stroškov, ki predstavljajo vrsto obremenitve podjetja, v to kategorijo pa spadajo tudi atributi, ki se uporabijo kot skupine za analizo pri analiziranju poslovanja. Ko uporabnik želi vnesti kategorijo stroška, se mu na prvi strani, prav tako kot pri vnosu uporabnika, prikažejo že vnesene kategorije (Slika 3.8), ki jih lahko uporabnik ureja ali izbriše. Na strani s pregledom že vnesenih kategorij je gumb »Dodaj«, ki uporabnika popelje na obrazce za vnos novih kategorij (Slika 3.9). Na obrazcu so tri vnosna polja, ki jih mora uporabnik izpolniti:

- Naziv: Naziv kategorije stroška, ki se ga nato uporabi pri vnos stroška, ki se ga upošteva pri mesečnem obračunu oziroma finančnem poročilu.

- Vrste stroška – Fiksni stroški: So enaki ne glede na obseg poslovanja. V primeru nedelovanja podjetja ti stroški pomenijo izgubo podjetju.

- Vrste stroška – Variabilni stroški: So stroški, ki so odvisni od obsega proizvodnje.

```
$( '#save' ).on('click',function(){
    var url = window.location.href;
    var id = $( '#category_id' ).val();
    var name = $( '#name' ).val();
    var fiks = $( '#fcost' ).val();
    var variabil = $( '#vcost' ).val();
    setCostsCategory(id, name, fiks, variabil);
});
```

Slika 3.10: Branje vnesenih vrednosti v JavaScript datoteki

Ko uporabnik shrani novo kategorijo stroška, se le-ta shrani v tabelo modul_costs_category, kamor se poleg naziva (name), fiksnih (fix_costs) ter variabilnih (variable_costs) stroškov, ki so predstavljeni kot vnos boolean (true ali false), vpiše še datum prvega vnosa (insert_datetime) ter datum posodobitve (update_datetime), ki je ob prvem vnosu enak datumu prvega vnosa, nato pa se v primeru urejanja kategorije spreminja le datum posodobitve. Ko uporabnik klikne na gumb »Shrani«, se izvede klic na javascript datoteko »stroski_kategorije.js«, v kateri preberemo vse vnesene vrednosti (Slika 3.10). Ko imamo vse vrednosti prebrane, kličemo funkcijo, ki s pomočjo AJAX-a (Slika 3.11) pošlje prebrane podatke v datoteko functions-admin.php. V datoteki functions-admin.php kličemo funkcijo `setCostsCategory()`, ki izvede sql-stavek UPDATE (Slika 3.12) v primeru, da kategorijo urejamo, v primeru novega vnosa pa s izvede INSERT sql-stavek.

Ko se poizvedba izvede, se uporabniški vmesnik vrne na prikaz vnesenih kategorij, kjer je že prikazan nov vnos. V primeru, da se uporabnik zmoti, lahko vnos popravi.

Po vnosu kategorij lahko uporabnik prične z vnašanjem stroškov za vpis v mesečno poročilo. To lahko stori na vmesniku (Slika 3.13) za vnos stroškov,

```
function setCostsCategory(_id, _name, _fiks, _variabil){

            $.ajax({
                    type: "POST",
                    dataType: "json",
                    url: "../backend/functions-admin.php",
                    data: {
                            action : "setCostsCategory",
                            id : _id,
                            name : _name,
                            fix_costs : _fiks,
                            variable_costs : _variabil,
                    },
                    success: function(data){
                        if(_mode == 'edit'){
                            window.location.href = '/admin/content/stroski-kategorije-uredi.php';
                        }else{
                            window.location.href = '/admin/content/stroski-kategorije.php';
                        }
                    },
                    error: function(e) {
                        if(_mode == 'edit'){
                            window.location.href = '/admin/content/stroski-kategorije-uredi.php';
                        }else{
                            window.location.href = '/admin/content/stroski-kategorije.php';
                        }
                    }
            });

}
```

Slika 3.11: Klic AJAX-a za prenos vnesenih vrednosti

v katerem so prikazani stroški, ki so razporejeni po mesecih in letih. Vnesene stroške je mogoče konec meseca izvoziti v xml-datoteko, ki jo lahko računovodja podjetja uporabi za vodenje računovodskih poslov. Ko se naredi zaključek vnosa mesečnih stroškov (možno jih je pregledovati po željenih datumih, vendar pa so zaključna poročila vedno od prvega do zadnjega dneva v mesecu), se nad temi stroški izvede analiza, ki da uporabniku vedeti, kateri stroški so za podjetje obetavni oziroma obremenjujoči. Uporabnik se na podlagi teh analiz lahko odloča, v katerem elementu podjetja so potrebne spremembe oziroma sankcije za izboljšanje rezultatov.

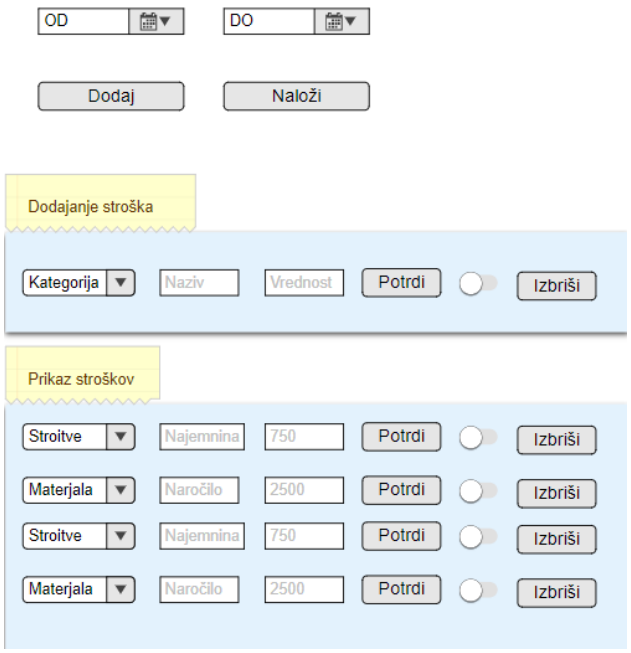
```php
function setCostsCategory(){
    global $db;

    $id = $_POST['id'];
    $name = $_POST['name'];
    $fixc = $_POST['fix_costs'];
    $variablec = $_POST['variable_costs'];

    $sql="UPDATE modul_costs_categories"
    . "SET name='".$name."', fix_costs='".$fixc."',"
    . "variable_costs='".$variablec."',"
    . "update_datetime=(CURRENT_TIMESTAMP) WHERE id = '".$id."';";

    $result=$db->query($sql);
}
```

Slika 3.12: SQL stavek za posodobitev vrednosti izbrane kategorije



Slika 3.13: Načrtovalna skica za vmesnik stroškov

Poglavje 4

Analize stroškov

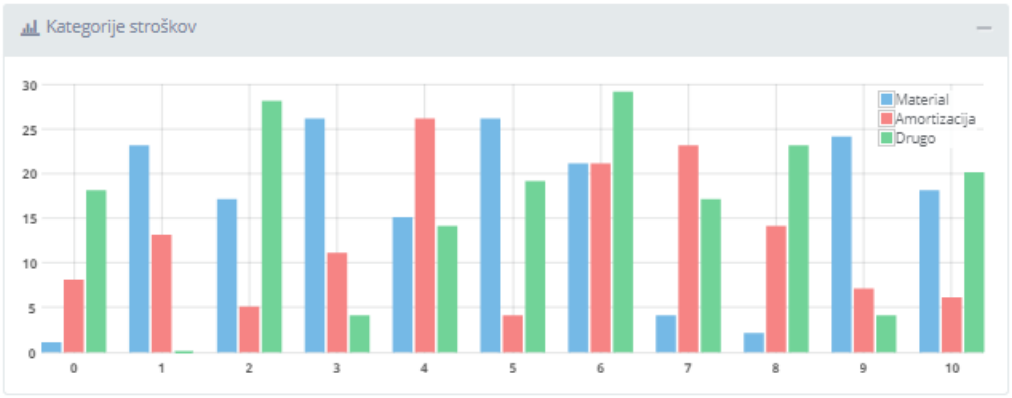
Namen modula je, da pomagamo uporabniku izboljšati poslovanje podjetja na eleganten ter preprost način. V ta namen je potrebno stroške podjetja vnesti v sistem ter jih nato sistemsko analizirati. V 3. poglavju smo spoznali, kako lahko uporabnik na preprost ter eleganten način vnaša stroške podjetja v sistem. V 4. poglavju pa bomo spoznali, kako te podatke analizirati ter predstaviti uporabniku na pravilen način.

4.1 Izračuni stroškov

4.2 Vmesnik

Vmesnik za analizo ponuja tri preglede nad poslovanjem:

- Pregled prodaje
- Pregled stroškov (Slika 4.1)
- Analiza poslovanja



Slika 4.1: Prikaz stroškov po kategorijah skozi določeno obdobje

4.2.1 Pregled prodaje

V vmesniku za pregled prodaje bo uporabniku omogočen pregled nad dohodki podjetja za izbrano obdobje. Modul za analizo se razvija kot dodatek oziroma dopolnitev obstoječi aplikaciji, v katero se vnaša prihodek podjetja. V modulu za vnos ter analizo poslovanja bo uporabniku omogočen splošen vpogled nad prihodki podjetja. Kot že omenjeno, lahko uporabnik izbere željeno obdobje, za katerega želi prikaz prihodkov podjetja. Prav tako izbere, za katero kategorijo artikla ter znamko želi izpis. V primeru, da izbira ostane prazna, se naredi izpis vseh prihodkov. V primeru, da je uporabnik lastnik trgovine, ima možnost izbire razvrstitve po uporabniku (izpis prihodkov, ki jih opravi določen prodajalec).

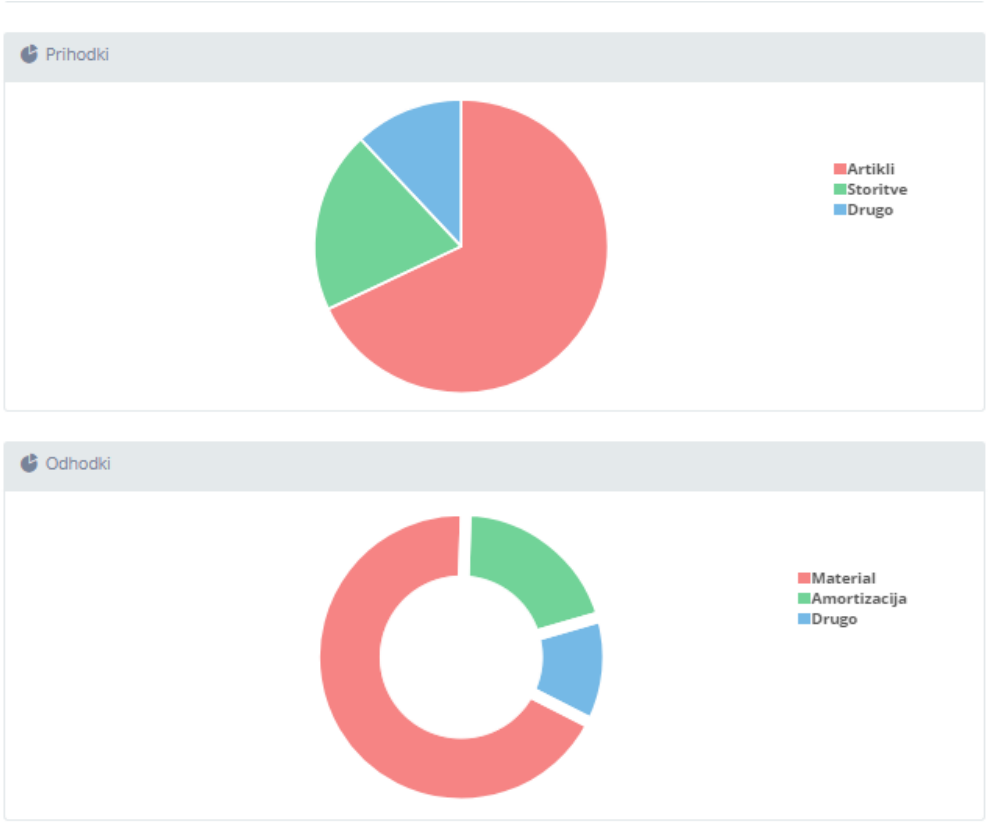
4.2.2 Pregled stroškov

V 3. poglavju smo opisali izgled ter delovanje vmesnika za vnašanje mesečnih stroškov poslovanja. Pregled stroškov je mogoč že pri vnašanju stroškov, vendar pa pregled ni preveč prijazen. V pregledu stroškov, ki ga omogoča vmesnik za analizo poslovanja, so stroški prav tako kot prodaja predstavljeni na uporabniku prijazen način. Stroške v poljubnem obdobju predstavimo s pomočjo grafov. Obstaja pa tudi možnost izbire kategorije stroškov, ki naj

se prikažejo posamično.

4.2.3 Analiza poslovanja

Pri pregledu analize poslovanja je prav tako mogoč pregled poslovanja podjetja glede na izbrano obdobje, kot pri zgoraj omenjenih pregledih. Možen bo tudi pregled posameznih kategorij stroškov ter artiklov. Nato pa bo grafično (Slika 4.2) predstavljena analiza poslovanja za izbrano obdobje oziroma za željeno kategorijo stroška.



Slika 4.2: Prikaz odhodkov in prihodkov v krožnem diagramu

4.3 Analiza

Za pravilen prikaz vnesenih stroškov ter njihovih izračunov je potrebno pravilno pristopiti do samega izračuna vrednosti. Pri izračunih največjo težavo predstavljajo stroški materiala, saj so cene spremenljive in niso v naprej fiksno določene (nasprotje so stroški storitev, kjer količino porabljenih storitev pomnožimo z njihovo ceno, saj pri storitvah ne držimo zalog). Take težave rešujemo z dodajanjem statusa oziroma indikatorja, ali je strošek fiksen ali spremenljiv. V primeru spremenljivih stroškov se moramo zavedati, da rezultat analize morda ne bo zanesljiv, saj lahko pričakujemo drugačne razlike med prihodki v naslednjih obračunskih obdobjih.

Pri analizi opravljamo primerjave med prihodki ter stroški podjetja. Zato je pomembno, da poznamo lastnosti stroškov za pravilen prikaz podatkov. V nasprotnem primeru aplikacija ne bi služila svojemu namenu. Glede na kategorijo stroška vzamemo prihodek, ki je posledica stroškov v analizi. Rezultat mora biti vedno pozitiven. V primeru negativnega rezultata uporabnik ugotovi, da mu stroški prinašajo negativno stanje pri poslovanju ter da je potrebno na tem področju uvesti spremembe. V stanju negativnega izračuna, je za reševanje področja že pozno, težave pa želimo reševati, preden pademo v izgube. V ta namen se uporabnike že prej opozarja ob preseženi, v naprej določeni vrednosti razlike med prihodkom ter odhodkom. Kljub negativni razliki pri neki kategoriji je lahko poslovanje še vedno pozitivno. Vendar pa uporabnikom želimo zagotoviti izboljšavo vodenja podjetja, kar pa pomeni, da je potrebno spremljati ter nadzorovati vse elemente poslovanja ter reševati težave, ko še niso prevelike.

Kategorija stroška amortizacije vsebuje majhno posebnost, zato jo bomo tudi podrobneje predstavili, saj je za pravilno analiziranje potrebno poznavanje kategorij.

4.3.1 Stroški amortizacije

Pri stroških amortizacije gre za stroške, za sredstva katerih je značilno, da pri poslovnem procesu izrabljajo oziroma trošijo njihove koristne lastnosti. Zaradi tehničnega in gospodarskega staranja postane vsako delovno sredstvo prej ali slej neuporabno. Tako lahko vsakemu delovnemu sredstvu rečemo, da ima dobo koristi. Ker se potroškov (količina potrošenega proizvodnega vira) delovnih sredstev ne da meriti v fizikalnih enotah, nabavo vrednosti razporedimo (amortiziranje) med stroške prek ocenjene dobe koristnosti. Amortizacija je torej strošek, ki nastane s prenašanjem nabavne vrednosti delovnega sredstva na poslovne učinke.

Osnova za izračun amortizacije je nabavna vrednost, ki jo pridobimo po naslednji formuli:

$$NV = NCS + SUS, \tag{4.1}$$

pri čemer je:

- NV – nabavna vrednost
- NCS – nakupna cena sredstva
- SUS – stroški usposobitve sredstva

V primeru, da sredstva ne izrabimo do konca njegove uporabne vrednosti, delovnem sredstvu ostane vrednost, ki ji pravimo ocenjena preostala vrednost. V tem primeru se amortizira le razlika med nabavno vrednostjo ter in ocenjeno preostalo vrednostjo. Tej razliki pravimo amortizljivi znesek.

$$AZ = NV + OPV, \tag{4.2}$$

kjer je:

- NV – nabavna vrednost
- AZ – amortizljivi znesek

- OPV – ocenjena preostala vrednost

Posebnost, ki je tudi razlog, zakaj predstavljamo izračun te kategorije stroška, je čas obračuna amortizacije. Obračun delovnega sredstva je šele prvi dan naslednjega meseca po vnosu stroška. S poznavanjem tega dejstva lahko pri analiziranju upoštevamo realne vrednosti. Pri analizi stroške amortizacije upoštevamo kot strošek, ki je potreben za delovanje podjetja (prevozna sredstva, računalniška oprema za administracijo ipd.).

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo se sprehodili skozi razvoj modula za nadzor ter analizo stroškov poslovanja, s katerim bodo lahko podjetja, ki še nimajo informacijskega sistema za nadzor poslovanja, na eleganten, preprost ter poceni način izboljšala poslovanje ter delovanje podjetja. Predstavili smo videz vmesnika, ki uporabniku ponuja preprosto uporabo modula, ter delovanje oziroma potek izvajanja kode v ozadju. Pogledali smo si tudi vrsto varnosti, ki jo je bilo treba vpeljati v program zaradi varnosti podatkov, ki ne smejo biti vidni oziroma na voljo vsem. Prav tako pa smo tudi predstavili vse uporabljene tehnologije ter se sprehodili skozi teorijo poslovnega poslovanja za lažjo predstavo, kaj se z modulom želi doseči.

Razvoj modula je še v razvoju ni dokončan ter je tako treba dokončati veliko malenkosti, tako pri videzu uporabniškega vmesnika kot tudi samem delovanju modula. Želja po zagotovitvi preproste uporabe, kjer program uporabnika ne obremenjuje z možnostmi, ki jih ne potrebuje, je izdelovanje programa otežilo. To težavo smo rešili z uporabo RBAC (angl. *Role-based access control*), ki prikazuje uporabniku elemente modula le, če so mu namenjeni. Tako ima lahko uporabnik le možnost pregleda vnesenih mesečnih stroškov, nikjer pa ga ne obremenjuje gumb za dodajanje, brisanje ali urejanje stroškov, če mu le to ni dovoljeno. Z omejitvijo dostopa smo učinkovito poskrbeli za varnost, kljub temu pa smo dodali zapisovanje vseh akcij upo-

rabnika za lažje odkrivanje napak, če do njih pride.

Uporabniški vmesnik smo oblikovali z vidika uporabnika, da mu zagotovimo dobro izkušnjo pri uporabi. Vendar pa bo na vmesniku kljub temu potrebno preoblikovanje, saj je v trenutnem stanju mogoč vnos le enega stroška. Elegantnejša rešitev bi bila, da lahko uporabnik vnese poljubno število stroškov, preden pritisne na gumb »Shrani«. Pav tako smo spoznali osem vrst stroškov, ki še niso bili v celoti vpeljani v sistem. Omenjene stroške je pomembno poznati pri sami analizi poslovanja, da znamo uporabniku na lep strukturiran način predstaviti poslovanje po vrstah stroškov ter ponuditi primerno smernico nadaljnjega poslovanja. Kljub pomembnosti zaključnih analiz je bila implementacija vnosa stroškov večji izziv, saj s tem uporabnik upravlja sam in ima z vnašanjem stroškov ter urejanjem največjo interakcijo s programom. Pri samih analizah je uporabniku le vizualno predstavljeno poslovanje izbranega meseca in z vmesnikom sam uporabnik le malo upravlja.

Pri vpeljavi analiz je bil zahtevnejši programerski del ter poznavanja teorije stroškov in postopka zaračunavanja le-teh. Da stranki prikažemo pravilne podatke, ki so za dobro poslovanje potrebne, je treba vedeti, kako računati stroške ter v katero kategorijo jih razvrstimo, da uporabniku s prvim vpogledom predstavimo stroške po pomembnih kategorijah in ne zahtevamo od uporabnika, naj jih razvrsti sam, saj bi tako zavili s poti do našega cilja.

V modul bi bilo smiselno vpeljati tudi možnost izvoza vnesenih stroškov v xml-datoteko, ki jo lahko računovodja podjetja uporabil za uvoz stroškov za namene računovodskega dela. S poznavanjem temeljnih poslovnih stroškov lahko olajšamo delo računovodju s kategoriziranjem stroškov pri izvozu, tako da s tem ne bi imel opravka. V prihodnosti se stremi k izdelavi sistema za računovodje, ki bo združljiv s sistemom, za katerega se razvija modul, katerega razvoj je predstavljen v diplomski nalogi.

Funkcionalnosti modula so zasnovane na podlagi modulov, ki so vsebovani v sistemih, kot sta SAP HANA [7] ter Microsoft Dynamics [4]. Vendar pa sta omenjena sistema zahtevnejša za uporabo, z našim izdelkom pa želimo

zajeti ciljno publiko, ki ji odgovarjajo preprostejši in cenejši sistemi.

Literatura

[1] Edmond Woychowsky. Creating Web Pages with Asynchronous JavaScript and XML. Dosegljivo: http://ptgmedia.pearsoncmg.com/images/9780132272674/downloads/0132272679_Woychowsky_book.pdf, Avgust 2006. [Dostopano: 28. 7. 2017].

[2] Lea Bevčič. Stroški v podjetju. Dosegljivo: https://mladipodjetnik.si/podjetniski-koticek/racunovodstvo. [Dostopano: 15. 7. 2017].

[3] Mario Lurig. PHP Reference: Beginner to intermediate PHP5. Dosegljivo: http://cdn.phpreferencebook.com/wp-content/uploads/2008/12/php_reference_-_beginner_to_intermediate_php5.pdf. [Dostopano: 1. 8. 2017].

[4] Microsof Dynamics. Dosegljivo: https://www.microsoft.com/en-us/dynamics365/home. [Dostopano: 11. 5. 2017].

[5] O'REILLY. *jQuery Cookbook*. O'Reilly Media, November 2009.

[6] Margaret Rouse. Role-based access control (RBAC). Technical report. [Dostopano: 15. 7. 2017].

[7] SAP HANA. Dosegljivo: https://www.sap.com/products/hana.html. [Dostopano: 12. 5. 2017].

[8] Osnove Twiga. Dosegljivo: https://learn.getgrav.org/themes/twig-primer. [Dostopano: 5. 8. 2017].

[9] Darja Urbas. Rezrvacije v Sloveniji in Nemčiji. Diplomska naloga, Fakulteta za ekonomijo, Univerza v Ljubljani, 2008.